

#include <SPI.h>

// Define ALTERNATE\_PINS to use non-standard GPIO pins for SPI bus

#ifdef ALTERNATE\_PINS

  #define VSPI\_MISO 2

  #define VSPI\_MOSI 4

  #define VSPI\_SCLK 0

  #define VSPI\_SS 33

  #define HSPI\_MISO 26

  #define HSPI\_MOSI 27

  #define HSPI\_SCLK 25

  #define HSPI\_SS 32

#else

  #define VSPI\_MISO MISO

  #define VSPI\_MOSI MOSI

  #define VSPI\_SCLK SCK

  #define VSPI\_SS SS

  #define HSPI\_MISO 12

  #define HSPI\_MOSI 13

  #define HSPI\_SCLK 14

  #define HSPI\_SS 15

#endif

#if CONFIG\_IDF\_TARGET\_ESP32S2

#define VSPI FSPI

#endif

**const** byte WAKEUP = 0b00000010; // Wake-up from standby mode

**const** byte STANDBY = 0b00000100; // Enter Standby mode

**const** byte RESET = 0b00000110; // Reset the device

**const** byte START = 0b00001000; // Start and restart (synchronize) conversions

**const** byte STOP = 0b00001010; // Stop conversion

**const** byte RDATAC = 0b00010000; // Enable Read Data Continuous mode (default mode at power-up)

**const** byte SDATAC = 0b00010001; // Stop Read Data Continuous mode

**const** byte RDATA = 0b00010010; // Read data by command; supports multiple read back

//Register Read Commands

**const** byte RREG = 0b00000000;

**const** byte WRET = 0b00000000;

**static** **const** **int** spiClk = 1000000; // 1 MHz

//uninitalised pointers to SPI objects

SPIClass \* vspi = NULL;

SPIClass \* hspi = NULL;

**void** setup() {

//initialise two instances of the SPIClass attached to VSPI and HSPI respectively

vspi = new SPIClass(VSPI);

hspi = new SPIClass(HSPI);

Serial.begin (115000);

//clock miso mosi ss

#ifndef ALTERNATE\_PINS

//initialise vspi with default pins

//SCLK = 18, MISO = 19, MOSI = 23, SS = 5

vspi->begin();

#else

//alternatively route through GPIO pins of your choice

vspi->begin(VSPI\_SCLK, VSPI\_MISO, VSPI\_MOSI, VSPI\_SS); //SCLK, MISO, MOSI, SS

#endif

#ifndef ALTERNATE\_PINS

//initialise hspi with default pins

//SCLK = 14, MISO = 12, MOSI = 13, SS = 15

hspi->begin();

#else

//alternatively route through GPIO pins

hspi->begin(HSPI\_SCLK, HSPI\_MISO, HSPI\_MOSI, HSPI\_SS); //SCLK, MISO, MOSI, SS

#endif

//set up slave select pins as outputs as the Arduino API

//doesn't handle automatically pulling SS low

pinMode(VSPI\_SS, OUTPUT); //VSPI SS

pinMode(HSPI\_SS, OUTPUT); //HSPI SS

delay(10); //delay to ensure connection

digitalWrite(VSPI\_SS, LOW); //Low to communicated

vspi->transfer(RESET);

digitalWrite(VSPI\_SS, HIGH); //Low to communicated

delay(10); //delay to ensure connection

}

// the loop function runs over and over again until power down or reset

**void** loop() {

//use the SPI buses

getDeviceID();

// hspiCommand();

delay(100);

}

**void** getDeviceID() {

// byte data = 0b01010101; // junk data to illustrate usage

// byte data = 0x20;

//use it as you would the regular arduino SPI API

vspi->beginTransaction(SPISettings(spiClk, MSBFIRST, SPI\_MODE1));

digitalWrite(VSPI\_SS, LOW); //pull SS slow to prep other end for transfer

vspi->transfer(0b00000010);

vspi->transfer(0b00100000);

vspi->transfer(0b00000001);

byte pom = vspi->transfer(0x00);

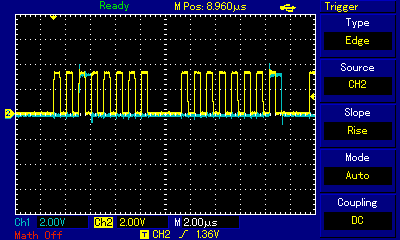
digitalWrite(VSPI\_SS, HIGH); //pull ss high to signify end of data transfer

vspi->endTransaction();

Serial.println(pom, BIN);

}

Ovo je kod kojeg sam koristio, to je službeni SPI kod koji se može naći pod primjerima SPI komunikacije u ARDUINO IDE, ja sam ga malo promijenio i ubacio konstante koje su meni potrebne.



Na ovoj slici se vidi da se naredbe pošalju. Šaljem naredbe prema uputama u datasheetu, opcode1 = 0b0010000 i opcode2 = 0b00000001. Prije toga šaljem naredbu wakeup. Probao sam i s njom i bez nje.

